Chapter 8

Computational Requirements

Optimal data assimilation has many advantages. Most importantly, we can derive truly best estimates, and we can assess whether the hydrologic model and our assumptions on its shortcomings are valid in the sense that they are statistically consistent with the assimilated data. Unfortunately, the computational burden for optimal data assimilation is also formidable. In this Chapter, we closely examine the computational requirements of the representer algorithm.

First, we review the computational demands of the synthetic experiments presented earlier and of a few more experiments that were not previously discussed (Section 8.1). Differences in the computational burden are explained and traced back to the design of the experiments. We then further investigate the computational requirements paying particular attention to the scalability of the approach (Section 8.2). This includes a review of the memory requirements and the possibility of parallel processing. Finally, we compare the computational requirements of the representer approach to other data assimilation methods (Section 8.3).

8.1 Computational Demands

In summary, the computational burden is influenced by three factors. First, the number of scalar data that are assimilated chiefly determines how many model integrations we need in order to derive the optimal estimates. Second, the number of nodes in the hydrologic model determines the CPU requirements for each model integration. Third, the computational effort is strongly influenced by the prior statistics and the number and length of the assimilation windows. The first two factors critically affect the scalability of the approach. We will postpone their discussion until Section 8.2. The last factors are the easiest to isolate, because it is straightforward to control the number of data and the size of the model. But they are also the hardest to understand (Section 8.1.2).

8.1.1 Computational Effort for the Synthetic Experiments

Table 8.1 summarizes the computational requirements of the experiments of Chapters 6 and 7 and of a few more experiments which were not presented earlier. The experiments are partitioned into three groups, according to the reference experiment they are based on. For the experiments that were presented earlier, we give the Section where a detailed description of the setup can be found. The other experiments are straightforward variations

Experiment			ideal	N_{ep}	N_{ot}	N_Z	k_{tl}	Σk_{cg}	$N_{\text{int}}^{\text{ind}}$	CPU [h]	Noise upd. [% CPU]	$N_{\text{int}}^{\text{ind}}/N_{\text{int}}^{\text{dir}}$ [%]
1	**Reference Exp. I**	Sec. 6.1	yes	512	12	6144	5	902	1819	30.5	3	3.0
2	(1:4) downscaling exp.	Sec. 6.3	yes	512	12	1536	3	334	677	11.8	3	7.3
3	(1:16) downscaling exp.	Sec. 6.3	yes	512	12	384	3	148	305	5.4	3	13.2
4	3-day repeat cycle	†	yes	512	4	2048	4	565	1142	20.4	3	7.0
5	6-day repeat cycle	†	yes	512	2	1024	3	365	739	13.2	3	12.0
6	Rain withheld	Sec. 7.2	no	512	12	6144	7	594	1209	20.9	3	1.4
7	Wrong texture	Sec. 7.3	no	512	12	6144	4	817	1646	28.5	3	3.3
8	**Reference Exp. II**	Sec. 6.2	yes	512	12	6144	4	439	890	18.0	25	1.8
9	(1:4) downscaling exp.	†	yes	512	12	1536	3	183	375	7.3	26	4.1
10	(1:16) downscaling exp.	†	yes	512	12	384	2	78	162	3.3	25	10.5
11	(1:64) downscaling exp.	†	yes	512	12	96	3	47	103	2.2	27	17.6
12	3-day repeat cycle	Sec. 6.4	yes	512	4	2048	4	290	592	11.5	26	3.6
13	6-day repeat cycle	Sec. 6.4	yes	512	2	1024	4	234	480	9.8	25	5.9
14	12-day repeat cycle	†	yes	512	1	512	4	136	284	5.6	25	6.9
15	3 assim. windows A‡	Sec. 7.1	no	512	12	6144	3.4	181	372	6.9	22	0.9
16	12 assim. windows B‡	Sec. 7.1	no	512	12	6144	2.8	45	98	2.4	18	0.3
17	12 assim. windows C‡	Sec. 7.1	no	512	12	6144	2.2	33	73	2.0	16	0.3
18	**Reference Exp. IIa**	†	yes	128	12	1536	3	197	403	1.9	20	4.4
19	(1:4) downscaling exp.	†	yes	128	12	384	3	99	207	0.9	22	8.9
20	(1:16) downscaling exp.	†	yes	128	12	96	2	36	78	0.4	21	20.0
21	3-day repeat cycle	†	yes	128	4	512	4	162	336	1.7	18	8.2
22	6-day repeat cycle	†	yes	128	2	256	3	94	197	0.9	22	12.8

†No other results from these experiments were presented. ‡Some entries are averages over the assim. windows.

Table 8.1: Computational effort for the synthetic experiments. The number of estimation pixels is denoted with N_{ep}, the number of observation times with N_{ot}, the number of scalar data with N_Z, the number of outer (tangent-linear) iterations with k_{tl}, and the total number of inner (conjugate-gradient) iterations with Σk_{cg}. For the number of model integrations we count all forward and backward runs separately. The total number of model integrations for the indirect representer method is $N_{\text{int}}^{\text{ind}} = 3k_{tl} + 2\Sigma k_{cg}$. The CPU time is measured on a DEC Alpha workstation with a clock speed of $333MHz$ and does not include I/O and preprocessing. We also indicate how much of the CPU time can be attributed to the process noise update. In the last column we compare the computational effort of the indirect representer method with the burden of the direct approach, which requires $N_{\text{int}}^{\text{dir}} = k_{tl}(2N_Z + 3)$ model runs.

on the ones presented throughout the thesis. For example, the repeat cycle experiments of the first group are designed in analogy to the repeat cycle experiments of Section 6.4, except that they are based on Reference Experiment I. Likewise, the downscaling experiments of the second group are designed similar to the ones in the first group but based on Reference Experiment II.

The experiments of the third group are variants of the second group at a coarser scale. For Reference Experiment IIa, we divide the same computational domain into $8 \times 16 = 128$ estimation pixels of $10km \times 10km$ each. The prior statistics are the same as in Reference Experiment II, except that we increase the correlation lengths by a factor of two (Section 8.2.2). Moreover, instead of spatially aggregating the true solution of Reference Experiment II, we use a different random seed when generating the synthetic true solution. This adds one more realization to the sparse body of data and helps corroborate the results.

For all experiments of Table 8.1, we indicate whether or not the assimilation conditions were ideal (Chapter 6). We then list the number of estimation pixels N_{ep}, the number of observation times N_{ot}, the number of scalar data N_Z, the number of outer (tangent-linear) iterations k_{tl}, and the total number of inner (conjugate-gradient) iterations Σk_{cg}, where the sum is over all outer iterations. We also show the number of model integrations $N_{\text{int}}^{\text{ind}}$ that are required in the indirect representer method. The last three columns contain the CPU time, the proportion of the CPU time attributed to the process noise update, and the percentage of the computational effort required by the indirect representer approach when compared to the direct method.

The total count for the indirect representer method is $N_{\text{int}}^{\text{ind}} = 3k_{tl} + 2\Sigma k_{cg}$ model integrations, where we count the forward and backward runs separately and equally. For each outer iteration, we have to compute the representer coefficients with the conjugate gradient solver in an inner loop (Section 2.3.4). During each of the iterations of the inner loop, we have to compute one linear combination of adjoint representer fields and one linear combination of state representer fields, which explains the factor of two in front of Σk_{cg}. In addition, for each outer iteration we need to calculate the prior trajectory (one integration) and, after obtaining the representer coefficients, we must finally solve the tangent-linear Euler-Lagrange equations for the state estimates (two integrations). This is why we need an additional $3k_{tl}$ model integrations.

By contrast, for the direct representer approach we need $N_{\text{int}}^{\text{dir}} = k_{tl}(2N_Z + 3)$ model integrations. During each outer iteration, we calculate the representer matrix by computing one prior trajectory, N_Z adjoint representers, and N_Z state representers. Finally, after obtaining the representer coefficients, we must calculate the estimated adjoint and state trajectories.

The CPU time is measured on a DEC Alpha workstation with a clock speed of $333MHz$. Note that the CPU times given in Table 8.1 do not include the preprocessing steps and input/output (I/O) operations. In the current implementation, all inputs and the process noise are kept in RAM, but the state (or the state representers) must be stored on disk. Depending on the type of experiment, the I/O operations increase the time for the assimilation by $5 \ldots 20\%$, where the higher number is for the experiments with twelve assimilation windows.

8.1.2 Computational Effort and Prior Statistics

If we compare the computational requirements of the experiments in the first and in the second group (Table 8.1), one obvious difference is the proportion of CPU time that goes into the process noise update. In Reference Experiment I, the model error is weak and the correlation length is only a fraction of the size of one estimation pixel. Consequently, we only spend about 3% of the CPU time on the process noise update, or approximately $2s$ of CPU time per model integration. In Reference Experiment II, the process noise variance is much stronger relative to the uncertainty in the initial condition, and, more importantly, the correlation length of the model error is increased fourfold. We now have to spend about 25% of the CPU time on the process noise update, which is equivalent to spending $18.5s$ per model run.

For small correlation lengths, we expect the burden for the process noise update per model integration to grow quadratically with the increase in the correlation length, because the convolution integral of the process noise update scales with the number of estimation pixels that are within one correlation length from any given point. This is compatible with the numbers given above if we also take boundary effects into account.

Moreover, the actual CPU requirements for the process noise update depend strongly on how many components we deem uncertain in the problem at hand. Considering that only one out of four components of the model equations is directly affected by process noise, and that the horizontal correlation length in Reference Experiment II is only 1.2 times the side of an estimation pixel, 25% is a significant share of the total computational effort. The share is so high because the model itself is not horizontally coupled and therefore extremely computationally efficient. The process noise update, by contrast, provides the horizontal coupling of the domain.

Despite the increased burden for the process noise, the computational effort for Reference Experiment II is still much less than for Reference Experiment I, even though the number of pixels and data is the same for both experiments. Whereas we need more than 30 hours of CPU time for Reference Experiment I, we only need 18 hours for Reference Experiment II. This can be explained by considering the condition number of the representer matrix.

Although we never explicitly compute the representer matrix, its condition number does of course affect the number of iterations needed in the conjugate gradient solver. In brief, the representer matrix encapsulates the correlation between the measurement predictions and the states. If there is very little model error, as was the case in Reference Experiment I, earlier and later brightness observations carry approximately the same information on the initial condition. As a consequence, the representer matrix is poorly conditioned. Since in Reference Experiment II the model error is strong, earlier and later brightness observations contain different amounts of information on the initial condition and on the model error at a given time. By increasing the model error variance and simultaneously decreasing the initial condition variance, we effectively improve the condition number of the representer matrix.

Another interesting comparison is between Reference Experiment I and the assimilation without precipitation data. Intriguingly, the computational effort for the latter experiment is only about two thirds of the computational burden of Reference Experiment I, although we do not supply the rain data to the assimilation algorithm. But by using different error statistics to compensate for the lack of precipitation data, we effectively improve the condition number of the representer matrix. Of course, we pay for the withholding of input data

and the increased convergence speed with a poorer estimate.

In summary, we can say that the prior statistics strongly influence the computational requirements. Increasing the horizontal correlation length of the model error leads to an increased computational burden for the process noise update. On the other hand, increasing the variance of the model error relative to the initial condition variance improves the condition number of the representer matrix and therefore eases the computational requirements.

8.1.3 Computational Effort for Multiple Assimilation Windows

We also observe a substantial decrease in the computational requirements if we go to multiple but shorter assimilation windows. Cutting the experiment period into three assimilation windows decreases the CPU time from 18 hours in Reference Experiment II to only 6.9 hours for experiment A for the entire two-week period. The savings are even more dramatic if we use twelve assimilation windows. For experiment B we only need 2.4 hours of CPU time. This decrease in CPU requirements is due to two factors.

First, we need relatively fewer model integrations as we increase the number of assimilation windows. For Reference Experiment II, we need 890 model runs to get the estimate, whereas in experiments A and B we only need on average 372 and 98, respectively. Second, the percentage of CPU time that goes into the process noise update decreases from 25% to 22% and 18%, respectively. This reflects the smaller relative importance of the model error compared to the initial condition uncertainty, because in experiments A and B we estimate the initial condition for each of the three or twelve assimilation windows. Although the reinitialization of the error covariances follows a naive scheme and the assimilation in these experiments is not strictly optimal, the estimates are nevertheless quite close to the optimal estimates of Reference Experiment II (Section 7.1). For a small sacrifice in optimality, the savings in computational effort are substantial enough to make assimilation intervals of a few days attractive in future applications.

8.2 Computational Effort and Scalability

For operational applications, we will have to tackle bigger problems, that is we must assimilate more data into larger domains. It is therefore of utmost importance to understand the scaling of the computational requirements as we increase the size of the problem. The two critical factors which determine the scaling are the number of scalar data that are assimilated and the number of nodes in the hydrologic model. In summary, the computational effort scales roughly with the product of the number of data and the number of estimation pixels, although the effort appears to grow somewhat less than linearly with the number of data. What follows is a more detailed discussion of this finding.

8.2.1 Number of Data and Efficiency of the Indirect Representer Method

The scaling of the computational effort with the number of data is closely related to the efficiency of the indirect representer approach. When we compare the computational burden of the indirect to the direct representer approach (Table 8.1), it is striking by how much the computational load is reduced in the indirect method. Typically, the effort for the indirect method is only a small percentage of what we would need in the direct approach. Of course, these huge savings come at a cost. The price is the ready availability of the posterior

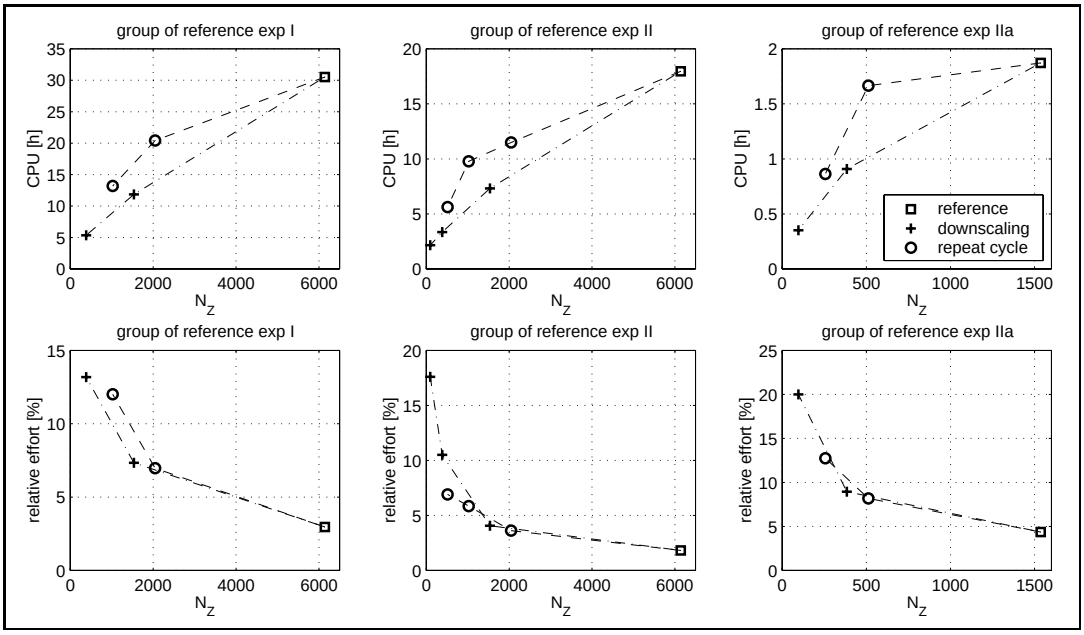

Figure 8.1: Computational effort for the ideal assimilation experiments. The top panels show the CPU requirements as a function of the number of scalar data that are assimilated. The lower panels show the computational effort of the indirect representer approach relative to the direct method. Obviously, there is redundancy in the data, and the computational burden grows less than linearly with the number of scalar data.

covariance information, which naturally comes with the direct method (Section 2.4). But this is a small price to pay, because it is certainly unnecessary to compute all of the posterior covariances at each outer iteration before the estimates are even dynamically consistent. We may still compute the individual representer fields after convergence has been achieved.

The efficiency of the indirect representer method does have an intuitive physical interpretation. In brief, there are many fewer degrees of freedom in the brightness images than there are scalar data, and the data are effectively compressed. To shed more light on this intuitive argument, it is helpful to investigate how the computational effort of the indirect representer method scales with the number of data. For reference, recall that the burden for the direct method scales linearly with the number of data (Section 8.1.1).

The top panels of Figure 8.1 show the CPU requirements for the three groups of synthetic experiments versus the number of scalar data. To avoid comparing apples and oranges, we only plot the ideal experiments of each group. For all three groups, the computational effort increases less than linearly with the number of data, suggesting that there is redundancy in the additional brightness observations. Another way to look at the same phenomenon is to plot the computational effort of the indirect method relative to the direct approach versus the number of data, which is done in the lower panels of Figure 8.1. We can see that the relative effort of the indirect method decreases as the number of data increases, again suggesting that there is relatively less information in the data as their number increases.

For the downscaling experiments, the result is obvious if we look at the spatial correlation scales of the dominant spatially distributed inputs, namely the soil texture and land cover

classes (Figure 5.3), the initial condition (Figure 6.3), and the precipitation. For all these inputs, the horizontal correlation scale is well above the scale of the estimation pixels, leading to redundant information in the brightness images. This finding confirms our hypothesis that the indirect representer approach and a priori data compression (Section 2.5) achieve related computational savings.

For the repeat cycle experiments, this result is not so obvious. From the 6-day repeat cycle experiment of Section 6.4 it is clear that an excellent estimate of the initial saturation can be obtained from only two brightness images at observation times 1 and 7. But the repeat cycle experiments based on Reference Experiment II have a strong model error which varies on a daily time scale. One would therefore expect that adding more information by assimilating more brightness images increases the computational effort proportionally. However, it appears that the brightness observations are effectively compressed in time as well as in space by the indirect representer method.

Unlike the spatial compression of individual images, such a compression in time is not straightforward, and it is not obvious whether it could be accommodated within the proposed data assimilation method by an a priori data compression scheme (Section 2.5). This means that the indirect representer approach is likely to be more efficient in reducing the computational effort than a priori data compression. But recall that a priori data compression greatly reduces the burden for calculating posterior error covariances (Section 2.5). Ideally, both schemes should be implemented.

Finally, it is interesting to take a quick look at the number of model integrations that are needed at each tangent-linear (outer loop) iteration to get the representer coefficients in the conjugate gradient solver. For Reference Experiment I, Figure 6.5 shows that $k_{cg} = 442$ (inner loop) iterations were necessary to derive the representer coefficients in the first (outer loop) iteration. In the remaining four tangent-linear (outer loop) iterations, on average only 115 such inner loop iterations were needed.

Similar results hold if we look at the other experiments (Figures 6.15, 6.18, 6.20, 7.5, and 7.11). On average, we only need about half the number of conjugate gradient iterations for the second and higher (outer) iterations than we do in the first (outer) iteration. This decrease in the computational burden is due to the much improved initial guess for the representer coefficients in the conjugate gradient solver. Obviously, we use the estimate for the representer coefficients from the previous iteration to initialize the conjugate gradient solver. This means that for nonlinear problems the indirect method is relatively more efficient (compared to the direct approach) than for linear problems.

8.2.2 Computational Effort, Resolution, and Size of the Domain

As we have seen above, the computational effort increases with the number of model runs that are required to solve for the estimate. One crucial factor determining the computational burden of the assimilation is therefore the time it takes to integrate the model once. Since the model is composed of a collection of independent vertical columns, the CPU requirements for each model integration will scale with the number of estimation pixels. Subtracting the effort for the model error update, each model integration covering the two-week experiment period takes approximately $60s$ of CPU time for the experiments of the first two groups with 512 estimation pixels. For the experiments of the third group with 128 estimation pixels, each model integration takes roughly $15s$. This translates into a CPU requirement of $0.01s$ per estimation pixel and per experiment day on the $333MHz$

DEC Alpha workstation.

Although the number of model integrations required for the estimate is mostly determined by the number of scalar data being assimilated, the number of nodes may affect the computational demand beyond merely increasing the CPU requirement for each model run. However, given all the other factors, in particular the complicated dependence of the computational burden on the prior statistics, it is hard to isolate the scaling of the computational demand with the number of nodes.

Suppose we increase the horizontal resolution of a given area, but we keep the horizontal correlation length for the model error fixed and we assimilate the same (coarse-scale) data. This will change the computational effort through at least three effects. First and trivially, each model integration will take longer at the fine resolution than at the coarse resolution. Second, one correlation length covers more estimation pixels at the fine resolution than at the coarse resolution, and the effort for the model error update will increase (Section 8.1.2). Third, the same(!) data may have different information content when viewed from the coarse and from the fine resolution of the estimation pixels, leading to different savings in the indirect representer method. This last effect is related to the increase in the relative computational effort of the indirect approach as we decrease the number of data by assimilating coarser brightness images (Section 8.2.1).

By comparing experiments with the same number of data from the second group and the third group (Table 8.1), we can check whether the computational effort scales only linearly with the number of estimation pixels. As mentioned above, we have increased the correlation length in Reference Experiment IIa and the related experiments with respect to Reference Experiment II. This makes the relative cost of the model error update approximately equal for both groups. If we now compare experiments of the two groups for which the same number of data are assimilated, we find that the CPU requirements increase approximately linearly with the number of estimation pixels. For example, if we divide the CPU time for experiment 9 by the CPU time for experiment 18, we get 3.9, while there are four times as many estimation pixels in experiment 9 than in experiment 18. For experiments 10 and 19, we get 3.7, and for experiments 14 and 21, we get 3.4. For experiments 11 and 20, on the other hand, we get 6.1, but this may be due to the extreme downscaling ratio of experiment 11. In summary, we have not found evidence that the computational effort increases more than linearly with the number of nodes.

8.2.3 Memory Requirements

The memory requirements for the assimilation algorithm are substantial. Ideally, all variables are kept in RAM (random-access memory) during the assimilation to keep the additional cost for I/O (input/output) operations low. As mentioned above, this was not possible with our computing equipment. But differences in speed aside, we can treat RAM and memory on the harddrive as being equivalent for the sake of this discussion. Also note that it is usually possible to trade off memory requirements against CPU time by recomputing dependent quantities each time they are needed rather than storing them.

We now discuss the memory requirements in detail. First, we have to store the model inputs. For the numerical parameters derived from the soil texture and land cover classes, the memory requirements are proportional to the number of estimation pixels and vertical nodes. For the micro-meteorologic inputs, the memory requirements are proportional to the length of the assimilation window. They are also proportional to the number pixels,

unless we opt for recomputing the interpolated fields whenever they are needed. Since during the assimilation the model must be run many times, it makes sense to keep the interpolated micro-meteorologic inputs at every pixel, together with some derived variables such as the land surface resistances, in RAM. In our implementation, the model inputs for the experiments of Chapters 6 and 7 require about $60MB$ in double precision.

Since the underlying hydrologic model is nonlinear, the adjoint operators depend on the state trajectories. Moreover, the variational method processes the entire assimilation window in a batch mode. It is therefore necessary to store the full state trajectory. In our example, the number of scalar states for all times is roughly 10^7 (Sections 4.2 and 5.2). For double precision, this is equivalent to about $80MB$.

Moreover, we need to store the estimates of the uncertain inputs. The memory requirements for the initial condition parameters are small, but we must also store up to two copies of the model error fields. First, the adjoint operator depends on the model error, because the coefficient in front of the process noise term in the state equation depends on the state itself (Section 2.2). We must therefore store the model error estimate of the previous iteration. Second, we may have to store the model error that is under update in the current iteration. Depending on the implementation, we may be able to use the memory allocated for the state trajectories to store the current model error estimate. But since we do not keep the entire state trajectory in RAM due to memory limitations, this is not possible in our implementation.

The actual memory requirements for storing the process noise depend strongly on how many components of the model we deem uncertain in the problem at hand. However, the memory requirements will always scale with the length of the assimilation window. For the experiments of Chapters 6 and 7, we need about $26MB$ of memory for storing the model error.

8.2.4 Parallel Computing

By design, the direct representer method is ideally suited for parallel computing [Bennett and Baugh, 1992]. In theory, each of the individual representer fields can be integrated simultaneously on a separate processor. The indirect representer approach, by contrast, does not generally allow for such parallel processing. But in our case, the structure of the hydrologic model would allow us to integrate the individual estimation pixels in parallel. Ideally, with N_{ep} processors, each model integration would only take as long as it takes to run a single one-dimensional column. Consequently, parallel computing offers great opportunities for the land surface data assimilation problem.

8.3 Comparison with Other Assimilation Techniques

In this Section, we briefly compare the computational requirements of the iterated indirect representer technique, the gradient-descent ("adjoint") method [Bennett, 1992], the full Kalman filter (KF) [Gelb, 1974], the Ensemble Kalman Filter (EnKF) (Section 1.2.2, [Evensen, 1994b]), error subspace statistical estimation (ESSE) (Section 1.2.2, [Lermusiaux and Robinson, 1999a]), and Optimal Interpolation (Section 1.2.1, [Daley, 1991]). Since we have only implemented the representer algorithm, the comparison with the other methods remains theoretical in nature. Nevertheless, an approximate count of the floating point operations provides sufficient insight into the characteristic requirements of each method.

Among the techniques mentioned above, the Kalman filter, the representer method, and the gradient-descent approach are optimal algorithms which fully propagate the error covariances. The big difference between these methods is that the Kalman filter propagates the error covariances explicitly, whereas the representer approach and the gradient-descent technique accomplish the error covariance propagation implicitly through the adjoint equations. The Ensemble Kalman Filter, ESSE, and Optimal Interpolation are suboptimal assimilation algorithms which rely on approximations of the dynamic evolution of the error covariances.

We now discuss the approximate operations count of each method for the experiments of Chapters 6 and 7. Recall from Sections 4.2 and 5.2 that the state vector at each time step is of length $N_x N_y (N_z + 5) = 6144$. With $N_t = 1280$ time steps, the total number of scalar states is $N_x N_y (N_z + 5) N_t \approx 10^7$. For each model integration, we simply count $\mathcal{O}(10^7)$ floating point operations. The dimension of the process noise in the example is $3 N_x N_y = 1536$ at every time step or a total of $3 N_x N_y N_t \approx 2 \cdot 10^6$ scalar variables. The number of data is 512 in each of the 12 images, and the total number of scalar data is $N_Z = 6144$. Note that we only discuss the leading order of the operations count.

8.3.1 Optimal Assimilation Algorithms

The explicit propagation of the error covariances in the Kalman filter requires manipulating matrices of the size of the state vector at each time step. The operations count for such matrix multiplications scales with the size of the matrix to the third power. Since this has to be done at every time step, the total count for the Kalman filter is $\mathcal{O}(10^{14})$. The actual operations count is even higher, because in the examples the process noise is colored, and the state must first be augmented. Note that solving the smoothing problem with the Kalman smoother approximately doubles the effort. The computational burden is so heavy because the Kalman filter comes with the full posterior covariance information.

As discussed above, the iterated indirect representer method scales with the time it takes to integrate the model once times the number of model integrations. Using 1000 model integrations and doubling the effort to accommodate the process noise update, the total operations count is $\mathcal{O}(10^{10})$. It is important to note that this does not include the posterior covariance information. We also stress that here we make use of the particular structure of the land surface model and the nature of the brightness observations, which implies that the results of this Section are valid only for the land surface data assimilation problem discussed in this thesis.

At first glance, the gradient-descent method requires only two model integrations at each iteration for the evaluation of the gradient. But the number of iterations needed for convergence strongly depends on the technique that is used for stepping down the gradient. The steepest-descent algorithm, which does not require any additional effort, is notoriously inefficient and should never be used [Press et al., 1992]. One alternative is to determine a "conjugate" gradient descent direction and to search for the minimum along this direction. For the line search, the model needs to be integrated multiple times and the objective function must be determined. If there is process noise, the latter step makes the method unattractive in practice, because the model error covariance is of size $10^6 \times 10^6$ and must be inverted. The problem becomes especially severe if the process noise is colored, as it is in our example. Regardless of the particular variant employed, the gradient-descent method is always impractical in our example because the search happens in the space of the uncertain

inputs, of which there are many more than there are scalar data.

8.3.2 Suboptimal Assimilation Algorithms

The effort for the Ensemble Kalman filter or related techniques, such as for example the error subspace statistical estimation (ESSE) scheme of Lermusiaux and Robinson [1999a], depends critically on the number of realizations or ensemble members that are used to approximate the error covariances. For a problem with 2800 data, a total of 10^8 states, and no model error, Lermusiaux and Robinson [1999b] need approximately 200 ensemble members. Moreover, the decomposition of the forecast error covariance and the actual update step require additional matrix manipulations. If we use their expression for the operations count (Table 2 of [Lermusiaux and Robinson, 1999b]) and substitute our numbers, the ESSE scheme and the EnKF require roughly $\mathcal{O}(10^{11})$ floating point operations. Since these techniques approximate the error covariance propagation, the estimates are not optimal, but the methods provide valuable information about the posterior error covariances.

Finally, Optimal Interpolation as the most suboptimal scheme essentially requires one model integration and the manipulation of matrices which are of the size of the data vector at each update time. The operations count for Optimal Interpolation is roughly $\mathcal{O}(10^9)$.

8.3.3 Summary

In conclusion, the full Kalman filter or smoother cannot be used for large-scale land data assimilation. Likewise, the optimal adjoint-based gradient-descent method is not computationally feasible if model errors are present. From a computational point of view, the indirect iterated representer technique is competitive with suboptimal sequential Monte Carlo methods like the Ensemble Kalman Filter and ESSE. The great advantage of the representer technique lies with the optimality of the estimates. If posterior covariance information is required, the sequential Monte Carlo methods may be a better choice. Optimal Interpolation is of course the cheapest alternative, but the estimates are poorer and no posterior covariance information is provided.